Application No. 09/727,744
Amendment dated October 14, 2005
Reply to Office Action dated July 28, 2005

Remarks/Arguments

Applicant has received and carefully reviewed the Office Action mailed July 28, 2005. Claims 1, 3-10, and 12-27 remain pending. Claims 2 and 11 were previously canceled, without prejudice. Reexamination and reconsideration are respectfully requested.

Allowed Claims

Applicants would like to thank the Examiner for indicating that claims 19-27 are allowed.

35 U.S.C. § 103(a) Rejections

In paragraph 4 of the Office Action, the Examiner rejected claims 1, 3, 5-10, 12-13 and 14-18 under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,338,136 to Col et al. (hereinafter Col) in view of U.S. Patent No. 3,577,189 to Cocke et al. (hereinafter Cocke). Applicants must respectfully disagree.

Claim 1 recites:

> 1.   (Currently Amended) A method for processing a conditional jump instruction in a pipelined instruction processor, the method comprising:
> generating at least one status bit based on a digital value to be stored, the at least one status bit relating to a particular condition of a conditional jump instruction and specifying if the particular condition of the conditional jump instruction is satisfied or not;
> storing the digital value and the at least one status bit to a commonly addressed memory location; and
> in response to a conditional jump instruction, reading from the commonly addressed memory location the digital value and the at least one status bit to resolve whether the condition of the conditional jump

11 of 15

Application No. 09/727,744
Amendment dated October 14, 2005
Reply to Office Action dated July 28, 2005

instruction is satisfied before the conditional jump instruction reaches an
arithmetic logic stage of the pipelines instruction processor.

As can be seen, claim 1 recites a method for processing a conditional jump instruction in
a pipelined instruction processor. The method includes a generating step, a storing step
and a reading/resolving step. The generating step generates at least one status bit <u>based
on a digital value to be stored</u>, wherein the at least one status bit relates to a particular
condition of a conditional jump instruction and specifies if the particular condition of the
conditional jump instruction is satisfied or not. The storing step stores the digital value
and the at least one status bit to a <u>commonly addressed memory location</u>. Finally, the
reading/resolving step, in response to a conditional jump instruction, reads from the
<u>commonly addressed memory location</u> the digital value and the at least one status bit to
resolve whether the condition of the conditional jump instruction is satisfied before the
conditional jump instruction reaches an arithmetic logic stage of the pipelines instruction
processor.

Col clearly does not disclose or suggest this method. As noted in the interview
summary mailed March 17, 2005, the status bits of Col are stored in the Flags register,
which is read by the condition evaluator shown in Figure 5B, element 572 in the store
stage of the pipeline. During operation, the ALU executes the conditional jump
instruction, which updates the Flags register, and the Flags register is read in turn by the
condition evaluator to determine whether the condition has been met or not.

As can be seen, Col does not appear to discloses or suggests generating at least
one status bit <u>based on a digital value to be stored</u>, and then storing the digital value and
the at least one status bit to a <u>commonly addressed memory location</u>, as recited in claim

12 of 15

Application No. 09/727,744
Amendment dated October 14, 2005
Reply to Office Action dated July 28, 2005

1. In fact, the Flags register of Col does not appear to store a digital value at all from

which the status bits are based, and more specifically, does not appear to store a digital

value from which the status bits are based along with the at least one status bits to a

commonly addressed memory location. Also, the status bits stored in the Flags register

do not appear to be written or stored to memory by Store Logic 571 or during the write

back logic 582. Also, and as acknowledged by the Examiner, Col does not appear to

teach or suggest resolving whether the condition of the conditional jump instruction is

satisfied before the conditional jump instruction reaches an arithmetic logic stage of the

pipelined instruction processor, as recited in claim 1.

The Examiner states that Cocke suggests resolving whether a condition of a

conditional jump instruction is satisfied before the conditional jump instruction reaches

the arithmetic logic stage of a pipelined instruction processor. The Examiner concludes

that a person of ordinary skill in the art at the time the invention was made would have

recognized that Cocke's method of branching improves performance, and therefore, to

incorporate the branching method of Cocke to improve processor performance.

After careful review, it does not appear that Cocke teach or suggest what Col

lacks. That is, Cocke does not appear to discloses or suggests generating at least one

status bit based on a digital value to be stored, and then storing the digital value and the at

least one status bit to a commonly addressed memory location. Instead, in Cocke, the

instruction includes an OP Code, which is provided to an instruction decoder 40 (see

Cocke, Figure 1A). Instruction decoder 40 decodes the OP Code field of the instruction

in Row 0, and if that instruction is found to be a branch instruction, the type of branch

instruction as seen in Table IV of Cocke is indicated by one of the output lines 42. The

13 of 15

Application No. 09/727,744
Amendment dated October 14, 2005
Reply to Office Action dated July 28, 2005

particular one of lines 42 that is activated will indicate the function of the two bits of a

condition register (C-register); the bits being specified by the I-field and J-field of the

instruction. (see, Cocke, column 7, lines 38-47; Figure 1A).

Referring now to Figure 2 of Cocke, which is a more detailed representation of

the Function Generator seen generally in Figure 1A. Lines 42, which are discussed

above, include lines 263, 264, 265,..., 270 and form gating inputs to the Function

Generator 62. The two pairs of lines 58, 60 from the C-register are also inputs to the

Function Generator. Lines 58 are indicative of the true and complement values of the

particular bit of the C-register that is selected by the I-Field of the instruction, while lines

60 are indicative of the true and complement values of the particular bit of the C-register

that is selected by the J-Field of the instruction. The Function Generator 62 also includes

output line 65 which indicates that the specified function of the selected bits of the C-

register has been computed as true, and the branch will be successful. Line 67 in Figure

2 is also provided which, when activated, indicates that the specified function of the two

bits selected by the I- and J-fields was found to be false and hence the branch will be

unsuccessful. (see, Cocke, column 7, lines 47-65).

The C-register, which is generally shown in Figure 1C of Cocke, is shown in

more detail in Figure 3. As noted above, the I- and J-fields of the instruction indicate

which bits of the C-register are to be inputted to the Function Generator. The C-register

includes a plurality of flip-flops (designated $C_0, C_1, C_2, ... C_{31}$ in Figure 3). Referring to

Figure 3, lines 58 comprise C_i and C_i-bar, indicative of the true and complement value,

respectively, of the contents of the bit of the C-register designated by the I-field of the

instruction in Row 0. Likewise, lines 60 are designated C_j and C_j-bar, indicative of the

Application No. 09/727,744
Amendment dated October 14, 2005
Reply to Office Action dated July 28, 2005

true and complement value, respectively, of the bit of the C-register indicated by the J-field of the instruction in Row 0. (see, Cocke, column 6, line 64 through column 7, line 5). The values of bits C_0, C_1, C_2, ... C_{31} are dependent upon the various machine conditions, according to the requirements of the system and are set by means within the system which do not form a part of the Cocke invention (see, Cocke, column 7, lines 34-37).

It is unclear what part of Cocke would correspond to the status bits of claim 1. According to claim 1, the at least one status bit relates to a particular condition of a conditional jump instruction and specifies if the particular condition of the conditional jump instruction is satisfied or not. In Cocke, the Op Code, I-field or J-field of the instruction in Row 0 would not appear to correspond to the at least one status bit of claim 1, as none of these fields appear to specify if the particular condition of the conditional jump instruction is satisfied or not. It could be argued that the values C_0, C_1, C_2, ... C_{31} of the C-register might correspond to the at least one status bit of claim 1. However, there is no indication in Cocke that these values are generated based on a digital value to be stored, as recited in claim 1. Instead, Cocke state that the values of bits C_0, C_1, C_2, ... C_{31} are dependent upon the various machine conditions, according to the requirements of the system and are set by means within the system which do not form a part of the Cocke invention (see, Cocke, column 7, lines 34-37).

Furthermore, Cocke does not appear to store the digital value and the at least one status bit to a commonly addressed memory location. Also, Cocke does not appear to teach, in response to a conditional jump instruction, reading from the commonly addressed memory location the digital value and the at least one status bit to resolve

15 of 15

Application No. 09/727,744
Amendment dated October 14, 2005
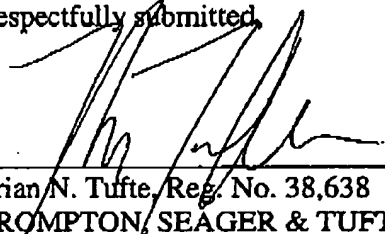Reply to Office Action dated July 28, 2005

whether the condition of the conditional jump instruction is satisfied before the

conditional jump instruction reaches an arithmetic logic stage of the pipelines instruction

processor.

As such, and as can readily be seen, Cocke does not appear to teach or suggest

what Col lacks. For these and other reasons, claim 1 is believed to be clearly patentable

over Col in view of Cocke. For similar and other reasons, dependent claims 3-9 are also

believed to be clearly patentable over Col in view of Cocke. For similar and other

reasons, independent claim 10 and dependent claims 12-18 are also believed to be clearly

patentable over Col in view of Cocke.

In view of the foregoing, it is believed that all pending claims 1, 2-10 and 12-27

are now in condition for allowance. Issuance of a notice of allowance in due course is

respectfully requested. If a telephone conference would be of assistance, please contact

the undersigned attorney at 612-359-9348.

Respectfully submitted,

Dated: _October 14, 2005_

Brian N. Tufte, Reg. No. 38,638
CROMPTON SEAGER & TUFTE, LLC
1221 Nicollet Avenue, Suite 800
Minneapolis, MN 55403-2402
Telephone: (612) 677-9050
Facsimile: (612) 359-9349